UNIT - 4
HIDDEN MARKOV MODELS

Sequential data- Markov models, HMM , Maximum likelihood for the HMM, The forward and backward algorithm, The sum-product algorithm, Scaling factors, Viterbi algorithm, Linear dynamical systems

Introduction to Sequential Data

- Sequential data are data points that occur in a specific order, often time-dependent.
- Examples: Daily rainfall measurements, Speech recognition (acoustic features in time frames), DNA nucleotide sequences

Important distinction:

- **Stationary distributions**: The generative process does not change over time.
- **Non-stationary distributions**: The process evolves over time.

- Previously, data points were treated as **independent and identically distributed (i.i.d.)**.
- For sequential data, this assumption does not hold, as nearby data points are often correlated.
- Example: Predicting whether it will rain tomorrow based on today's weather (i.i.d. model lacks this connection).

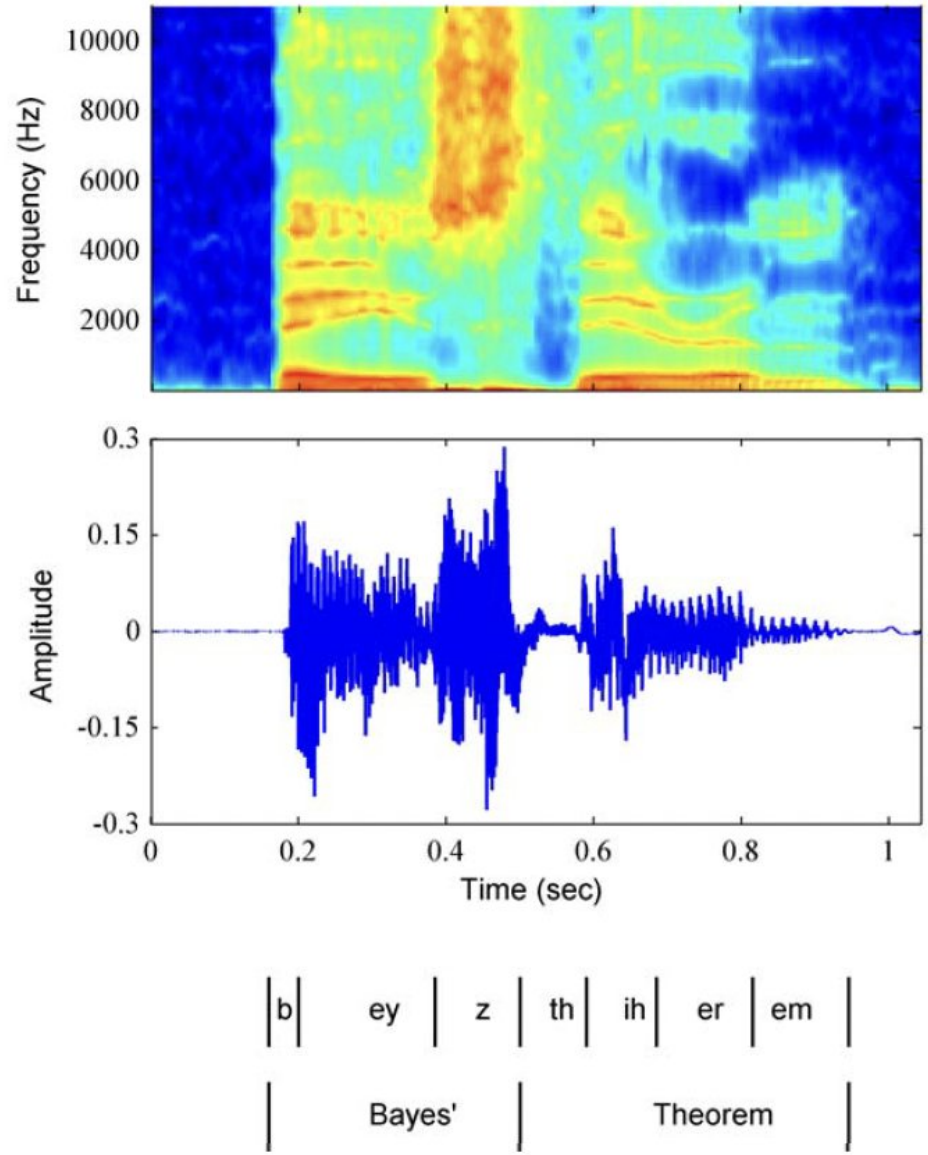


Figure 13.1 Example of a spectrogram of the spoken words "Bayes' theorem" showing a plot of the intensity of the spectral coefficients versus time index.

Markov Models

- We define the Markov process as a simple stochastic process in which the distribution of future states depends only on the present state and not on how it arrived in the present state.
- A random sequence has the Markov property if its distribution is determined solely by its current state.
- Any random process having this property is called a Markov random process. For observable state sequences (state is known from data), this leads to a Markov chain model. For non-observable states, this leads to a Hidden Markov Model (HMM).

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_N) = \prod_{n=1}^{N} p(\mathbf{x}_n | \mathbf{x}_1, \ldots, \mathbf{x}_{n-1}).$$

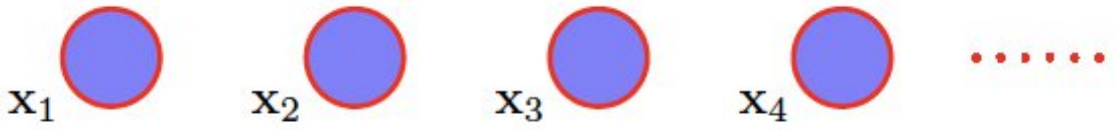


Figure 13.2 The simplest approach to modelling a sequence of observations is to treat them as independent, corresponding to a graph without links.

First Order Markov Chain

- In a **first-order Markov chain**, the current state depends only on the previous state
- First-order Markov chain is described as each observation being independent of all previous observations except the most recent.
- Example: Predicting weather based on whether it rained the day before.

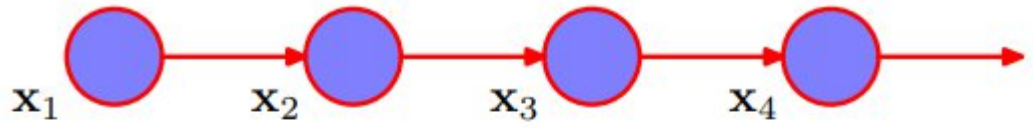


Figure 13.3 A first-order Markov chain of observations $\{\mathbf{x}_n\}$ in which the distribution $p(\mathbf{x}_n|\mathbf{x}_{n-1})$ of a particular observation $\mathbf{x}_n$ is conditioned on the value of the previous observation $\mathbf{x}_{n-1}$.

joint distribution for a sequence of N observations under this model is given by

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_N) = p(\mathbf{x}_1) \prod_{n=2}^{N} p(\mathbf{x}_n|\mathbf{x}_{n-1}). \tag{13.2}$$

Higher Order Markov Chain

- **Second-order Markov chain**: Depends on the last two observations
- More flexibility but with increased model complexity.

The joint distribution is now given by

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_N) = p(\mathbf{x}_1)p(\mathbf{x}_2|\mathbf{x}_1) \prod_{n=3}^{N} p(\mathbf{x}_n|\mathbf{x}_{n-1}, \mathbf{x}_{n-2}).$$

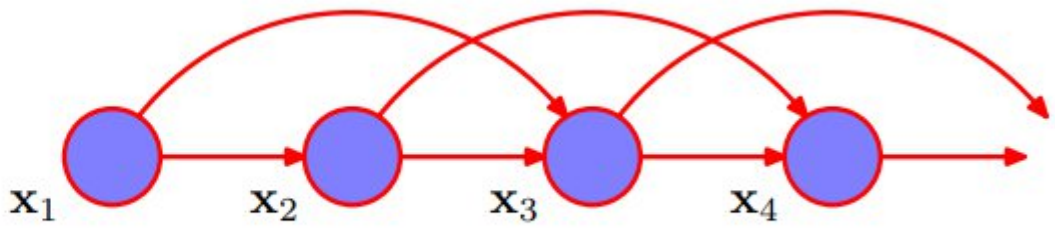


Figure 13.4 A second-order Markov chain, in which the conditional distribution of a particular observation $\mathbf{x}_n$ depends on the values of the two previous observations $\mathbf{x}_{n-1}$ and $\mathbf{x}_{n-2}$.

Markov Model

- A discrete (finite) system can be described as consisting of:
  - N distinct states.
  - Begins (at time t=1) in some initial state(s).
  - At each time step (t=1,2,…) the system moves from current to next state (possibly the same as the current state) according to transition probabilities associated with current state.
- This kind of system is called a finite, or discrete Markov model. The model has been named after Andrei Andreyevich Markov (1856 -1922)

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_N, \mathbf{z}_1, \ldots, \mathbf{z}_N) = p(\mathbf{z}_1) \left[\prod_{n=2}^{N} p(\mathbf{z}_n | \mathbf{z}_{n-1}) \right] \prod_{n=1}^{N} p(\mathbf{x}_n | \mathbf{z}_n).$$

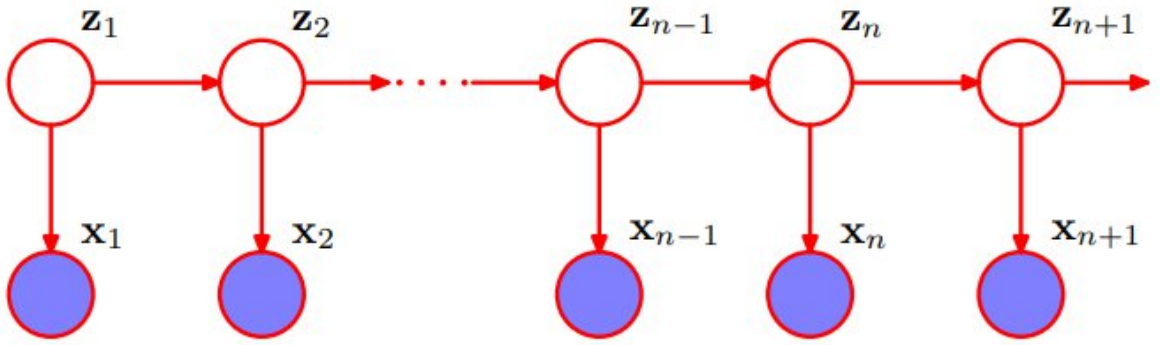


Figure 13.5 We can represent sequential data using a Markov chain of latent variables, with each observation conditioned on the state of the corresponding latent variable. This important graphical structure forms the foundation both for the hidden Markov model and for linear dynamical systems.

Hidden Markov Models

- A statistical model called a Hidden Markov Model (HMM) is used to describe systems with changing unobservable states over time.
- It is predicated on the idea that there is an underlying process with concealed states, each of which has a known result.
- Probabilities for switching between concealed states and emitting observable symbols are defined by the model.
- MMs are used in a wide range of industries, including finance, bioinformatics, and speech recognition.
- HMMs are useful for modelling dynamic systems and forecasting future states based on sequences that have been seen because of their flexibility.

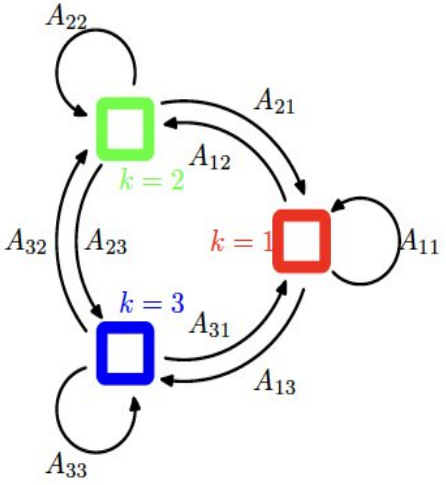
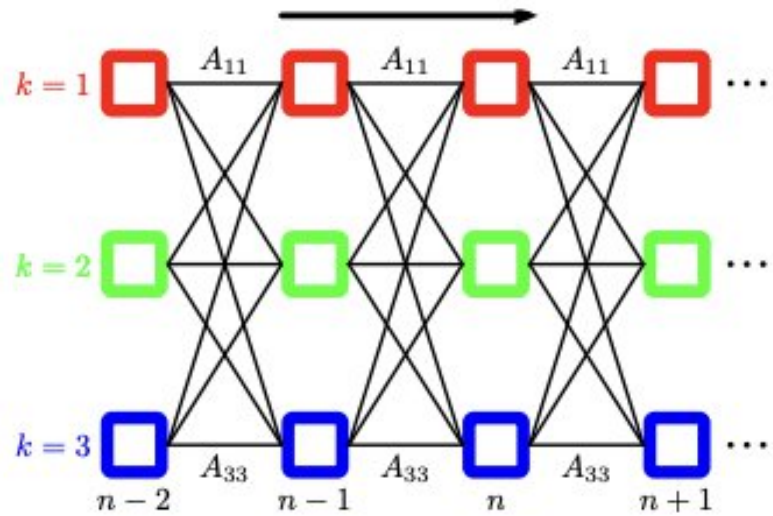


Figure 13.6 Transition diagram showing a model whose latent variables have three possible states corresponding to the three boxes. The black lines denote the elements of the transition matrix A_{jk}.

- Transition probability

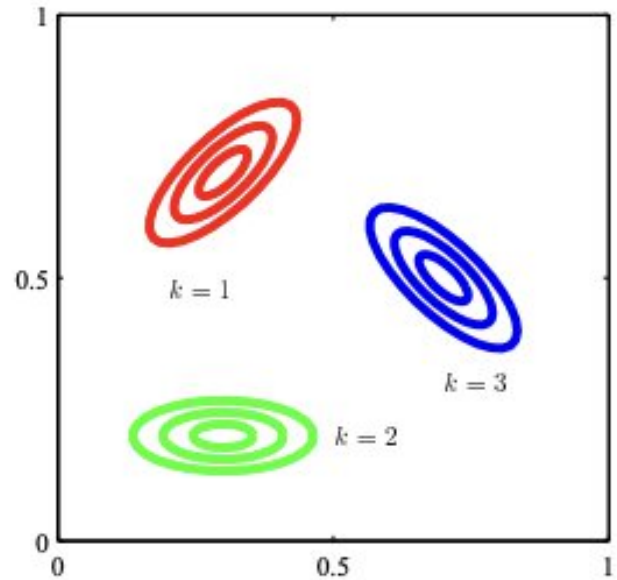
$$p(\mathbf{z}_n|\mathbf{z}_{n-1},\mathbf{A}) = \prod_{k=1}^{K}\prod_{j=1}^{K} A_{jk}^{z_{n-1,j}z_{nk}}$$

$$A_{jk} \equiv p(z_{nk}=1|z_{n-1,j}=1),$$
$$0 \leq A_{jk} \leq 1 \text{ and } \sum_k A_{jk} = 1$$



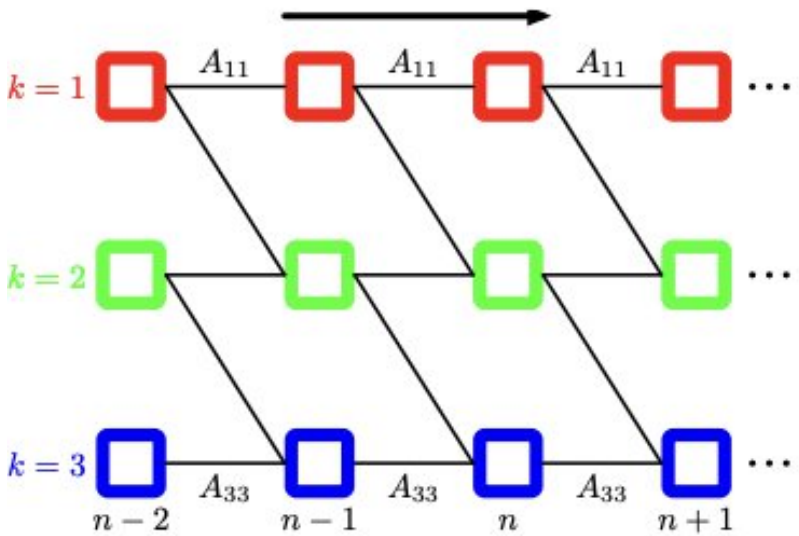
- Emission probability

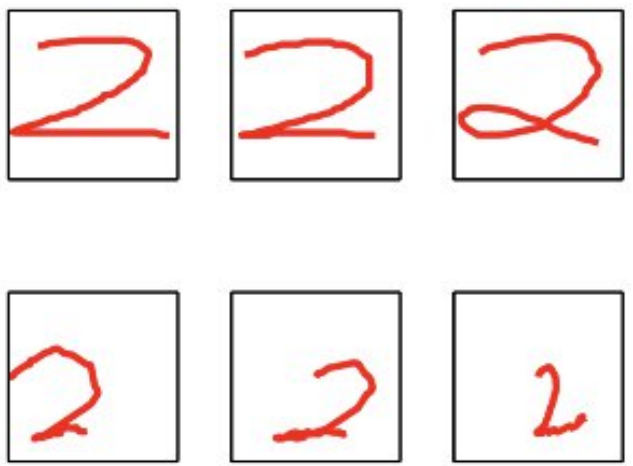
$$p(\mathbf{x}_n|\mathbf{z}_n,\phi) = \prod_{k=1}^{K} p(\mathbf{x}_n|\phi_k)^{z_{nk}}$$



HMM applications

- Speech recognition
- Natural language modelling
- Analysis of biological sequences (e.g. proteins and DNA)
- On-line handwriting recognition; Example: Handwritten digits
  - Left-to-right architecture
  - On-line data: each digit represented by the trajectory of the pen as a function of time





Maximum likelihood for the HMM

- We have observed a data set

$$\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\},$$

- so we can determine the parameters of an HMM

$$\theta = \{\pi, \mathbf{A}, \phi\}$$

  by using maximum likelihood.
- The likelihood function is

$$p(\mathbf{X}|\theta) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\theta)$$

Expectation maximization algorithm (EM)

- Initial selection for the model parameters: θ^{old}
- E step:
  - Posterior distribution of the latent variables $p(\mathbf{Z}|\mathbf{X}, \theta^{\text{old}})$

$$Q(\theta, \theta^{\text{old}}) = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \theta^{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z}|\theta) \qquad (13.12)$$

E step:

$$Q(\theta, \theta^{\text{old}}) = \sum_{k=1}^{K} \gamma(z_{1k}) \ln \pi_k + \sum_{n=2}^{N} \sum_{j=1}^{K} \sum_{k=1}^{K} \xi(z_{n-1,j}, z_{nk}) \ln A_{jk}$$

$$+ \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma(z_{nk}) \ln p(\mathbf{x}_n | \phi_k) \qquad (13.17)$$

- The marginal posterior distribution of a latent variable γ and the joint posterior distribution of two successive latent variables ξ

$$\gamma(\mathbf{z}_n) = p(\mathbf{z}_n | \mathbf{X}, \theta^{\text{old}}) \qquad (13.13)$$

$$\xi(\mathbf{z}_{n-1}, \mathbf{z}_n) = p(\mathbf{z}_{n-1}, \mathbf{z}_n | \mathbf{X}, \theta^{\text{old}}) \qquad (13.14)$$

M step:

- Maximize $Q(\theta, \theta^{\text{old}})$ with respect to parameters $\theta = \{\pi, \mathbf{A}, \phi\}$, treat $\gamma(\mathbf{z}_n)$ and $\xi(\mathbf{z}_{n-1}, \mathbf{z}_n)$ as constant. By using Lagrange multipliers

$$\pi_k = \frac{\gamma(z_{1k})}{\sum_{j=1}^{K} \gamma(z_{1j})} \tag{13.18}$$

$$A_{jk} = \frac{\sum_{n=2}^{N} \xi(z_{n-1,j}, z_{nk})}{\sum_{l=1}^{K} \sum_{n=2}^{N} \xi(z_{n-1,j}, z_{nl})} \tag{13.19}$$

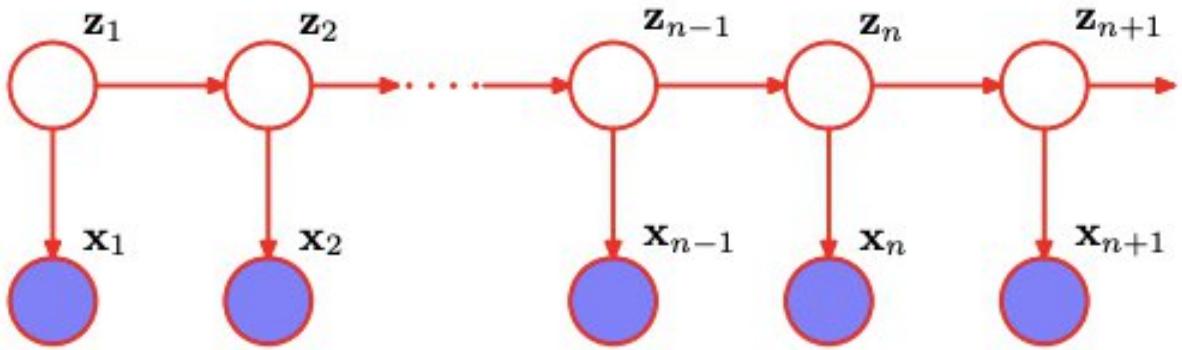
M step:

- Parameters ϕ_k independent
  $\rightarrow$ for Gaussian emission densities $p(\mathbf{x}|\phi_k) = \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)$

$$\mu_k = \frac{\sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n}{\sum_{n=1}^{N} \gamma(z_{nk})} \tag{13.20}$$

$$\Sigma_k = \frac{\sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \mu_k)(\mathbf{x}_n - \mu_k)^{\mathsf{T}}}{\sum_{n=1}^{N} \gamma(z_{nk})} \tag{13.21}$$

The forward-backward algorithm



- Two-stage message passing algorithm
- Several variants, we focus on alpha-beta algorithm

Evaluate $\gamma(\mathbf{z}_n)$

- Using Bayes' theorem

$$\gamma(\mathbf{z}_n) = p(\mathbf{z}_n|\mathbf{X}) = \frac{p(\mathbf{X}|\mathbf{z}_n)p(\mathbf{z}_n)}{p(\mathbf{X})} \tag{13.32}$$

$$= \frac{p(\mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{z}_n)p(\mathbf{x}_{n+1}, \ldots, \mathbf{x}_N|\mathbf{z}_n)}{p(\mathbf{X})}$$

$$= \frac{\alpha(\mathbf{z}_n)\beta(\mathbf{z}_n)}{p(\mathbf{X})} \tag{13.33}$$

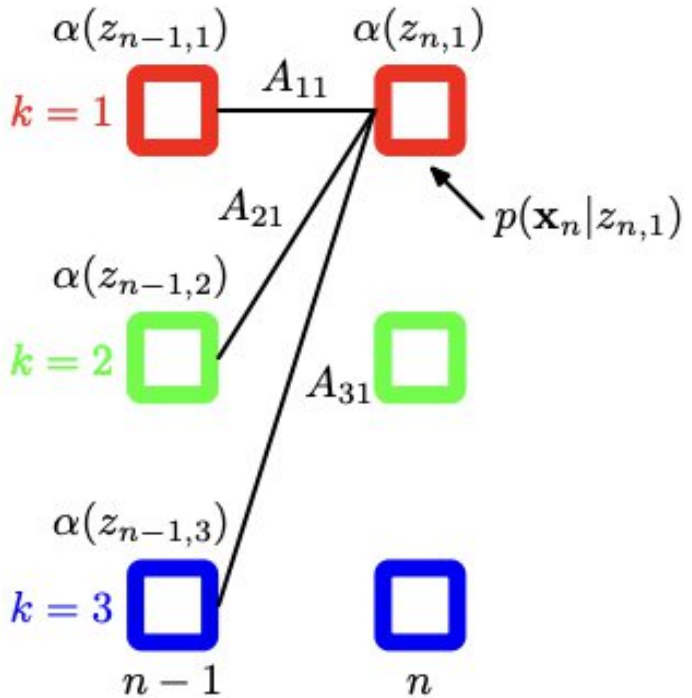
- where we have defined

$$\alpha(\mathbf{z}_n) = p(\mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{z}_n) \tag{13.34}$$

$$\beta(\mathbf{z}_n) = p(\mathbf{x}_{n+1}, \ldots, \mathbf{x}_N|\mathbf{z}_n) \tag{13.35}$$

Evaluate $\gamma(\mathbf{z}_n)$: Forward- Backward algorithm

Forward recursion for $\alpha(\mathbf{z}_n)$

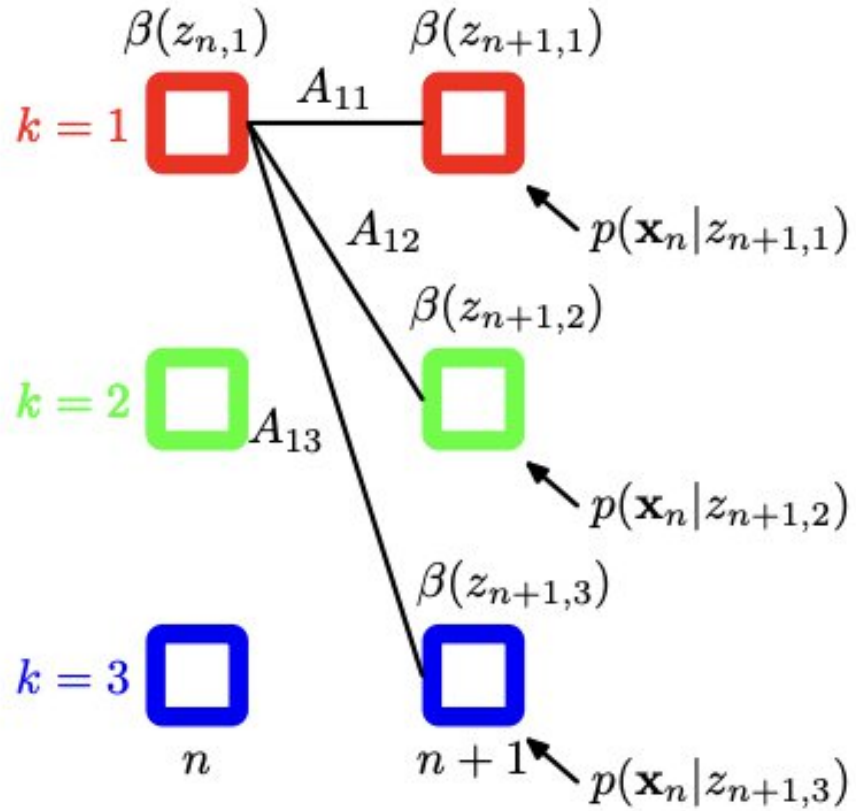


$$\alpha(\mathbf{z}_n) = p(\mathbf{x}_n|\mathbf{z}_n) \sum_{\mathbf{z}_{n-1}} \alpha(\mathbf{z}_{n-1}) p(\mathbf{z}_n|\mathbf{z}_{n-1}) \tag{13.36}$$

$$\alpha(\mathbf{z}_1) = p(\mathbf{x}_1, \mathbf{z}_1) = p(\mathbf{z}_1) p(\mathbf{x}_1|\mathbf{z}_1) = \prod_{k=1}^{K} \{\pi_k p(\mathbf{x}_1|\phi_k)\}^{z_{1k}} \tag{13.37}$$

Evaluate $\gamma(\mathbf{z}_n)$: Forward - Backward algorithm

Backward recursion for $\beta(\mathbf{z}_n)$



$$\beta(\mathbf{z}_n) = \sum_{\mathbf{z}_{n+1}} \beta(\mathbf{z}_{n+1}) p(\mathbf{x}_{n+1}|\mathbf{z}_{n+1}) p(\mathbf{z}_{n+1}|\mathbf{z}_n) \qquad (13.38)$$

$$\beta(\mathbf{z}_N) = 1$$

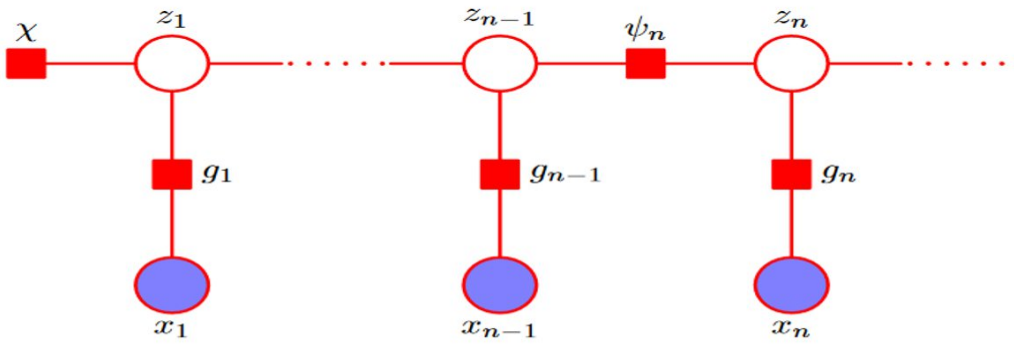
Evaluate $\xi(\mathbf{z}_{n-1}, \mathbf{z}_n)$

- Using Bayes' theorem

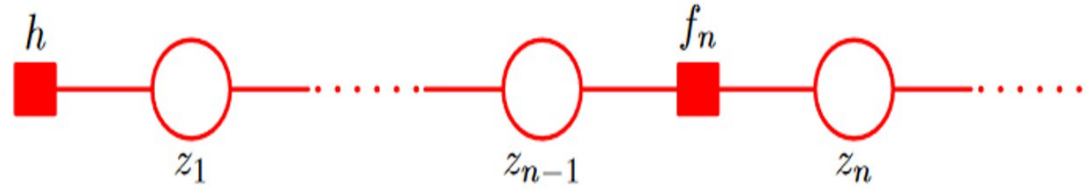
$$\begin{aligned}\xi(\mathbf{z}_{n-1}, \mathbf{z}_n) &= p(\mathbf{z}_{n-1}, \mathbf{z}_n|\mathbf{X}) \\ &= \frac{p(\mathbf{X}|\mathbf{z}_{n-1}, \mathbf{z}_n)p(\mathbf{z}_{n-1}, \mathbf{z}_n)}{p(\mathbf{X})} \\ &= \frac{\alpha(\mathbf{z}_{n-1})p(\mathbf{x}_n|\mathbf{z}_n)p(\mathbf{z}_n|\mathbf{z}_{n-1})\beta(\mathbf{z}_n)}{p(\mathbf{X})} \qquad (13.43)\end{aligned}$$

The sum-product algorithm for the HMM

- Applied to Hidden Markov Models (HMMs) for computing marginal distributions of hidden states.
- This algorithm allows for efficient computation by breaking down the problem into manageable pieces.
- The sum-product algorithm uses factor graphs to represent the dependencies between variables, allowing for efficient message passing.



Factor Graph



Simplified Factor Graph

- Start by converting the directed graph into a factor graph. The factor graph shows all variables explicitly, both latent and observed.
- For solving the inference problem, we condition on the observed variables $x_1, x_2, \ldots, x_N$.
- Absorb the emission probabilities into the transition probability factors.
- The transition probability factors, now simplified, include both the original transition and emission probabilities.

$$
\begin{aligned}
h(\mathbf{z}_1) &= p(\mathbf{z}_1)p(\mathbf{x}_1|\mathbf{z}_1) \\
f_n(\mathbf{z}_{n-1}, \mathbf{z}_n) &= p(\mathbf{z}_n|\mathbf{z}_{n-1})p(\mathbf{x}_n|\mathbf{z}_n).
\end{aligned}
$$

- Alpha-Beta Algorithm Derivation: Start by denoting the final hidden variable z_N as the root node.
- Message Passing: First, pass messages from the leaf node h to the root node.
- Message Propagation: Use the general results for message propagation.

$$\mu_{\mathbf{z}_{n-1}\to f_n}(\mathbf{z}_{n-1}) = \mu_{f_{n-1}\to \mathbf{z}_{n-1}}(\mathbf{z}_{n-1})$$

$$\mu_{f_n\to \mathbf{z}_n}(\mathbf{z}_n) = \sum_{\mathbf{z}_{n-1}} f_n(\mathbf{z}_{n-1},\mathbf{z}_n)\mu_{\mathbf{z}_{n-1}\to f_n}(\mathbf{z}_{n-1})$$

- Form of Messages: In the Hidden Markov Model, the propagated messages follow the structure derived from these general results

Recursion for $f \to \mathbf{z}$ the messages of the form:

$$\mu_{f_n \to \mathbf{z}_n}(\mathbf{z}_n) = \sum_{\mathbf{z}_{n-1}} f_n(\mathbf{z}_{n-1}, \mathbf{z}_n) \mu_{f_{n-1} \to \mathbf{z}_{n-1}}(\mathbf{z}_{n-1}).$$

$$\alpha(\mathbf{z}_n) = \mu_{f_n \to \mathbf{z}_n}(\mathbf{z}_n)$$

Messages that are propagated from the root node back to the leaf node

$$\mu_{f_{n+1} \to f_n}(\mathbf{z}_n) = \sum_{\mathbf{z}_{n+1}} f_{n+1}(\mathbf{z}_n, \mathbf{z}_{n+1}) \mu_{f_{n+2} \to f_{n+1}}(\mathbf{z}_{n+1})$$

$$\beta(\mathbf{z}_n) = \mu_{f_{n+1} \to \mathbf{z}_n}(\mathbf{z}_n)$$

- The sum-product algorithm also specifies how to evaluate the marginals once all the messages have been evaluated

$$p(\mathbf{z}_n, \mathbf{X}) = \mu_{f_n \to \mathbf{z}_n}(\mathbf{z}_n)\mu_{f_{n+1} \to \mathbf{z}_n}(\mathbf{z}_n) = \alpha(\mathbf{z}_n)\beta(\mathbf{z}_n)$$

$$\gamma(\mathbf{z}_n) = \frac{p(\mathbf{z}_n, \mathbf{X})}{p(\mathbf{X})} = \frac{\alpha(\mathbf{z}_n)\beta(\mathbf{z}_n)}{p(\mathbf{X})}$$

Scaling Factor

- In Hidden Markov Models (HMMs), the forward-backward algorithm suffers from **numerical underflow** because the forward variable $\alpha(\mathbf{z}_n)$ involves multiplying small probabilities, causing values to shrink exponentially.

- In (13.34), we defined $\alpha(\mathbf{z}_n) = p(\mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{z}_n)$ representing the joint distribution of all the observations up to $\mathbf{x}_n$ and the latent variable $\mathbf{z}_n$. Now we define a normalized version of α given by

$$\widehat{\alpha}(\mathbf{z}_n) = p(\mathbf{z}_n|\mathbf{x}_1, \ldots, \mathbf{x}_n) = \frac{\alpha(\mathbf{z}_n)}{p(\mathbf{x}_1, \ldots, \mathbf{x}_n)}$$

In order to relate the scaled and original alpha variables, we introduce scaling factors defined by conditional distributions over the observed variables

$$c_n = p(\mathbf{x}_n|\mathbf{x}_1, \ldots, \mathbf{x}_{n-1}).$$

From the product rule, we then have

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \prod_{m=1}^{n} c_m$$

and so

$$\alpha(\mathbf{z}_n) = p(\mathbf{z}_n|\mathbf{x}_1, \ldots, \mathbf{x}_n)p(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \left(\prod_{m=1}^{n} c_m\right)\widehat{\alpha}(\mathbf{z}_n).$$

We can then turn the recursion equation for α into one for $\widehat{\alpha}$ given by

$$c_n\widehat{\alpha}(\mathbf{z}_n) = p(\mathbf{x}_n|\mathbf{z}_n)\sum_{\mathbf{z}_{n-1}}\widehat{\alpha}(\mathbf{z}_{n-1})p(\mathbf{z}_n|\mathbf{z}_{n-1}).$$

We can similarly define re-scaled variables $\widehat{\beta}(\mathbf{z}_n)$ using

$$\beta(\mathbf{z}_n) = \left(\prod_{m=n+1}^{N} c_m\right)\widehat{\beta}(\mathbf{z}_n)$$

which will again remain within machine precision because the quantities $\widehat{\beta}(\mathbf{z}_n)$ are simply the ratio of two conditional probabilities

$$\widehat{\beta}(\mathbf{z}_n) = \frac{p(\mathbf{x}_{n+1}, \ldots, \mathbf{x}_N | \mathbf{z}_n)}{p(\mathbf{x}_{n+1}, \ldots, \mathbf{x}_N | \mathbf{x}_1, \ldots, \mathbf{x}_n)}.$$

The recursion result for β then gives the following recursion for the re-scaled variables

$$c_{n+1}\widehat{\beta}(\mathbf{z}_n) = \sum_{\mathbf{z}_{n+1}} \widehat{\beta}(\mathbf{z}_{n+1}) p(\mathbf{x}_{n+1} | \mathbf{z}_{n+1}) p(\mathbf{z}_{n+1} | \mathbf{z}_n).$$

In applying this recursion relation, we make use of the scaling factors c_n that were previously computed in the α phase

$$p(\mathbf{X}) = \prod_{n=1}^{N} c_n.$$

Similarly, using the given equations together we see that the required marginals are given by

$$\begin{aligned}\gamma(\mathbf{z}_n) &= \widehat{\alpha}(\mathbf{z}_n)\widehat{\beta}(\mathbf{z}_n) \\ \xi(\mathbf{z}_{n-1}, \mathbf{z}_n) &= c_n \widehat{\alpha}(\mathbf{z}_{n-1}) p(\mathbf{x}_n|\mathbf{z}_n) p(\mathbf{z}_n|\mathbf{z}_{-1}) \widehat{\beta}(\mathbf{z}_n).\end{aligned}$$

The Viterbi algorithm

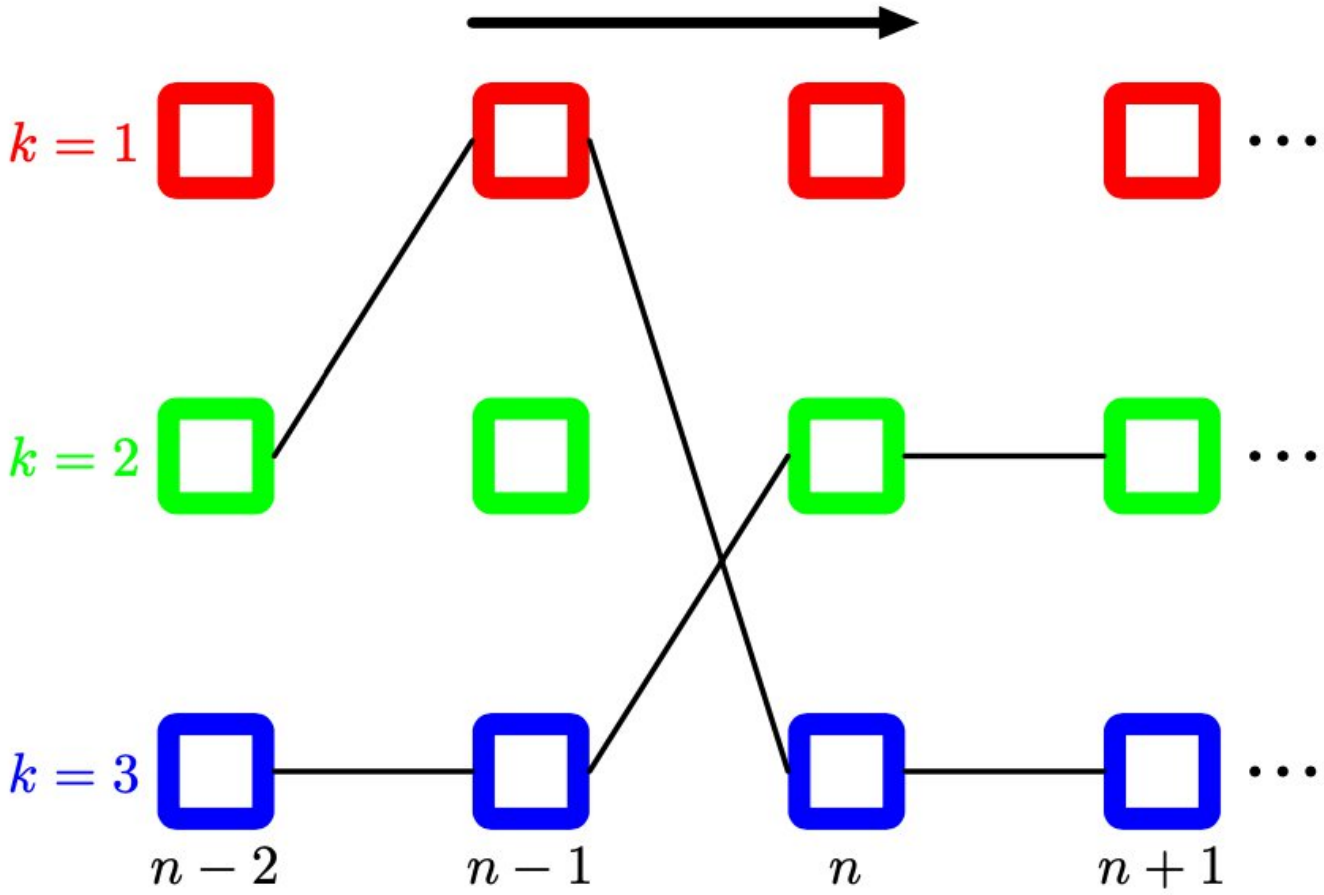
The Viterbi algorithm is used to find the **most probable sequence of hidden states** in Hidden Markov Models (HMMs) for a given observation sequence. It efficiently identifies the best path through the state space, crucial in applications like speech recognition.

- Finding the most probable sequence is different from maximizing individual state probabilities. The latter can lead to sequences with zero probability due to zero transition probabilities.

- The Viterbi algorithm employs the max-sum algorithm, working with log probabilities, which eliminates the need for re-scaling.

- HMMs can be represented as a factor graph, and messages are passed from leaf nodes to the root.

- A fragment of the HMM lattice showing two possible paths. The Viterbi algorithm efficiently determines the most probable path from amongst the exponentially many possibilities.
- For any given path, the corresponding probability is given by the product of the elements of the transition matrix A_{jk} corresponding to the probabilities $p(\mathbf{z}_{n+1}|\mathbf{z}_n)$ for each segment of the path, along with the emission densi $p(\mathbf{x}_n|k)$ associated with each node on the path.



Max-Sum Message Passing

$$\begin{aligned}\mu_{\mathbf{z}_n \to f_{n+1}}(\mathbf{z}_n) &= \mu_{f_n \to \mathbf{z}_n}(\mathbf{z}_n) \\ \mu_{f_{n+1} \to \mathbf{z}_{n+1}}(\mathbf{z}_{n+1}) &= \max_{\mathbf{z}_n} \left\{ \ln f_{n+1}(\mathbf{z}_n, \mathbf{z}_{n+1}) + \mu_{\mathbf{z}_n \to f_{n+1}}(\mathbf{z}_n) \right\}.\end{aligned}$$

Recursion for Messages

$$\omega(\mathbf{z}_{n+1}) = \ln p(\mathbf{x}_{n+1}|\mathbf{z}_{n+1}) + \max_{\mathbf{z}_n} \{\ln p(\mathbf{x}_{+1}|\mathbf{z}_n) + \omega(\mathbf{z}_n)\}$$

Initialization of Messages

$$\omega(\mathbf{z}_1) = \ln p(\mathbf{z}_1) + \ln p(\mathbf{x}_1|\mathbf{z}_1).$$

Maximization of Joint Distribution

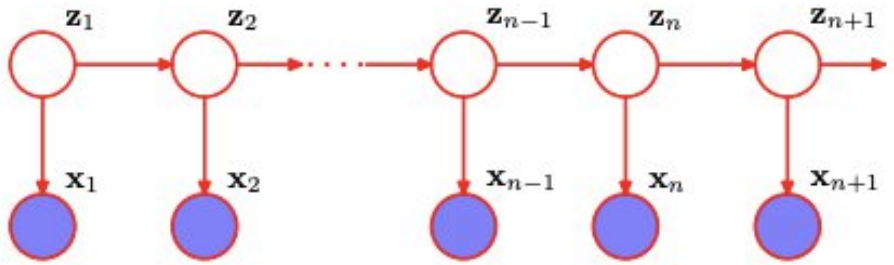
$$\omega(\mathbf{z}_n) = \max_{\mathbf{z}_1, \ldots, \mathbf{z}_{n-1}} p(\mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{z}_1, \ldots, \mathbf{z}_n).$$

Backtracking to Find Path

$$k_n^{\max} = \psi(k_{n+1}^{\max}).$$

The Viterbi algorithm efficiently computes the most probable sequence of hidden states by maintaining only the best paths, reducing the computational cost from exponential to linear with respect to the length of the sequence.

Linear Dynamical Systems



A linear-Gaussian model

- The general form of algorithms for the LDS are the same as for the HMM
- Continuous latent variables
- Both observed $\mathbf{x}_n$ and latent $\mathbf{z}_n$ variables Gaussian
  - Joint distribution over all variables, marginals and conditionals are Gaussian

  $\Rightarrow$ The sequence of individually most probable latent variable values is the same as the most probable latent sequence (no Viterbi considerations)

- Transition and emission probabilities

$$p(\mathbf{z}_n|\mathbf{z}_{n-1}) = \mathcal{N}(\mathbf{z}_n|\mathbf{A}\mathbf{z}_{n-1}, \Gamma) \tag{13.75}$$
$$p(\mathbf{x}_n|\mathbf{z}_n) = \mathcal{N}(\mathbf{x}_n|\mathbf{C}\mathbf{z}_n, \Sigma) \tag{13.76}$$

- The initial latent variable

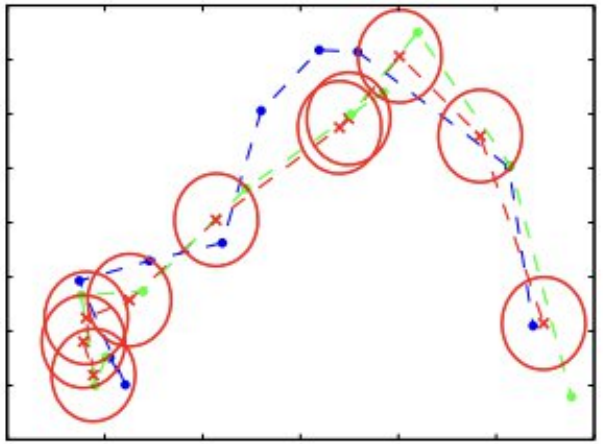
$$p(\mathbf{z}_1) = \mathcal{N}(\mathbf{z}_1|\mu_0, \mathbf{V}_0) \tag{13.77}$$

- The parameters $\theta = \{\mathbf{A}, \Gamma, \mathbf{C}, \Sigma, \mu_0, \mathbf{V}_0\}$ determined using maximum likelihood through EM

Inference in LDS

1. Find the marginal distributions for the latent variables conditional on the observation sequence
2. Given the parameters $\theta = \{\mathbf{A}, \Gamma, \mathbf{C}, \Sigma, \mu_0, \mathbf{V}_0\}$, predict the next latent state $\mathbf{z}_{n+1}$ and next observation $\mathbf{x}_{n+1}$

- Sum-product algorithm
  - Kalman filter (forward-recursion, α message)
  - Kalman smoother (backward-recursion, β message)
- Application of the Kalman filter: tracking



- True positions of the object
- Noisy measurements of the positions
- x Means of the inferred positions

Learning in LDS

- Determine $\theta = \{\mathbf{A}, \Gamma, \mathbf{C}, \Sigma, \mu_0, \mathbf{V}_0\}$ using maximum likelihood (again)
- Expectation maximization
  - E step:

$$Q(\theta, \theta^{\text{old}}) = \mathbb{E}_{\mathbf{Z}|\theta^{\text{old}}}[\ln p(\mathbf{X}, \mathbf{Z}|\theta)] \qquad (13.109)$$

  - M step: Maximize with respect to the components of θ

Extensions of LDS

- The marginal distribution of the observed variables is Gaussian
  - $\Rightarrow$ use *Gaussian mixture* as the initial distribution for $\mathbf{z}_1$
- Make Gaussian approximation by linearizing around the mean of the predicted distribution
  - *Extended Kalman filter*
- Combining the HMM with a set of linear dynamical systems
  - *Switching state space model*

Sampling methods

- Needed for dynamical systems which do not have a linear-Gaussian
- Sampling-importance-resampling formalism
  $\Rightarrow$ a sequential Monte Carlo as the particle filter
- Particle filter algorithm:
  At time step n
  - obtained a set of samples and weights
  - observe $\mathbf{x}_{n+1}$
  - evaluate samples and weights for time step $n+1$